

[illegible]

```

TTTTTTTTT1  EEEEEEEEE  MM      MM  PPPPPPP  LL      AAAAAA  TTTTTTTTTT  EEEEEEEEE
TTTTTTTTTT  EEEEEEEEE  MM      MM  PPPPPPP  LL      AAAAAA  TTTTTTTTTT  EEEEEEEEE
TT          EE          MMMM  MMMM  PP      PP  LL      AA      AA  TT          EE
TT          EE          MMMM  MMMM  PP      PP  LL      AA      AA  TT          EE
TT          EE          MM  MM  MM  PP      PP  LL      AA      AA  TT          EE
TT          EEEEEEEE  MM      MM  PPPPPPP  LL      AA      AA  TT          EEEEEEEE
TT          EEEEEEEE  MM      MM  PPPPPPP  LL      AA      AA  TT          EEEEEEEE
TT          EE          MM      MM  PP      PP  LL      AAAAAAAAAA  TT          EE
TT          EE          MM      MM  PP      PP  LL      AAAAAAAAAA  TT          EE
TT          EE          MM      MM  PP      PP  LL      AA      AA  TT          EE
TT          EEEEEEEEE  MM      MM  PP      PP  LL      AA      AA  TT          EEEEEEEEE
TT          EEEEEEEEE  MM      MM  PP      PP  LL      AA      AA  TT          EEEEEEEEE
LL          LL          SSSSSSSS
LL          IIIIII      SSSSSSSS
LL          IIII       SS
LL          II        SS
LL          II        SS
LL          II        SS
LL          II        SSSSSS
LL          II        SSSSSS
LL          II        SS
LL          II        SS
LL          II        SS
LL          II        SS
LLLLLLLLLL  IIIIII      SSSSSSSS
LLLLLLLLLL  IIIIII      SSSSSSSS

```

```

1 0001 0 MODULE TEMPLATE (
2 0002 0 IDENT = 'V04-000',
3 0003 0 ADDRESSING_MODE(EXTERNAL=GENERAL,
4 0004 0 NONEXTERNAL=LONG_RELATIVE)
5 0005 0 ) =
6 0006 1 BEGIN
7 0007 1
8 0008 1
9 0009 1 *****
10 0010 1 *
11 0011 1 * COPYRIGHT (c) 1978, 1980, 1982, 1984 BY
12 0012 1 * DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
13 0013 1 * ALL RIGHTS RESERVED.
14 0014 1 *
15 0015 1 * THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
16 0016 1 * ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE
17 0017 1 * INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER
18 0018 1 * COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
19 0019 1 * OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY
20 0020 1 * TRANSFERRED.
21 0021 1 *
22 0022 1 * THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
23 0023 1 * AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
24 0024 1 * CORPORATION.
25 0025 1 *
26 0026 1 * DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
27 0027 1 * SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
28 0028 1 *
29 0029 1 *****
30 0030 1
31 0031 1
32 0032 1 ++
33 0033 1 FACILITY: VAX/VMS MONITOR Utility
34 0034 1
35 0035 1 ABSTRACT:
36 0036 1
37 0037 1 The TEMPLATE module contains the routines to create
38 0038 1 templates for the various display screens.
39 0039 1
40 0040 1 ENVIRONMENT:
41 0041 1
42 0042 1 Unprivileged, user mode.
43 0043 1
44 0044 1 AUTHOR: Henry M. Levy , CREATION DATE: 28-April-1977
45 0045 1
46 0046 1 MODIFIED BY:
47 0047 1
48 0048 1 V03-012 TLC1072 Thomas L. Cafarella 17-Apr-1984 11:00
49 0049 1 Add volume name to DISK display.
50 0050 1
51 0051 1 V03-011 TLC1066 Thomas L. Cafarella 01-Apr-1984 11:00
52 0052 1 Add SYSTEM class.
53 0053 1
54 0054 1 V03-010 TLC1060 Thomas L. Cafarella 12-Mar-1984 11:00
55 0055 1 Make multi-file summary work for homogeneous classes.
56 0056 1
57 0057 1 V03-009 TLC1054 Thomas L. Cafarella 07-Mar-1984 11:00

```



```

58 0058 1
59 0059 1
60 0060 1
61 0061 1
62 0062 1
63 0063 1
64 0064 1
65 0065 1
66 0066 1
67 0067 1
68 0068 1
69 0069 1
70 0070 1
71 0071 1
72 0072 1
73 0073 1
74 0074 1
75 0075 1
76 0076 1
77 0077 1
78 0078 1
79 0079 1
80 0080 1
81 0081 1
82 0082 1
83 0083 1
84 0084 1
85 0085 1
86 0086 1
87 0087 1
88 0088 1
89 0089 1
90 0090 1

```

```

Fix positioning of data lines for homogeneous classes.

V03-008 PRS1006 Paul R. Senn 17-FEB-1984 14:00
Add support for "computed" items

V03-008 TLC1052 Thomas L. Cafarella 17-Feb-1984 11:00
Add multi-file summary capability.

V03-007 PRS1005 Paul R. Senn 13-JAN-1983 10:00
Allow flexible spacing between screen items

V03-006 SPC0006 Stephen P. Carney 01-Jul-1983 09:00
Change some RWxxx (resource wait state) codes.

V03-005 TLC1035 Thomas L. Cafarella 06-Jun-1983 15:00
Add homogeneous class type and DISK class.

V03-004 TLC1028 Thomas L. Cafarella 14-Apr-1983 16:00
Add interactive user interface.

V03-004 SPC0001 Stephen P. Carney 25-Mar-1983 15:00
Add RWxxx and MUTEX states in place of MWAIT state.

V03-003 TLC1020 Thomas L. Cafarella 1-Jul-1982 15:00
Remove semi-colon to eliminate BLISS INFO message.

V03-002 TLC1010 Thomas L. Cafarella 29-Mar-1982 15:00
Eliminate lower-case "a" strings from summary bar graphs.

V03-001 TLC1005 Thomas L. Cafarella 25-Mar-1982 17:00
Alter vertical spacing for classes with 13 items.

```

```

92 0091 1
93 0092 1
94 0093 1
95 0094 1
96 0095 1
97 0096 1
98 0097 1
99 0098 1
100 0099 1
101 0100 1
102 0101 1
103 0102 1
104 0103 1
105 0104 1
106 0105 1
107 0106 1
108 0107 1
109 0944 1
110 1244 1
111 1245 1
112 1246 1
113 1247 1
114 1248 1
115 1249 1
116 1250 1
117 1251 1
118 1252 1
119 1253 1
120 1254 1
121 1255 1
122 1256 1
123 1257 1
124 1258 1
125 1259 1
126 1260 1
127 1261 1
128 1262 1
129 1263 1
130 1264 1
131 M 1265 1
132 1266 1
133 1267 1
134 1268 1
135 1269 1
136 1270 1
137 M 1271 1
138 M 1272 1
139 1273 1
140 1274 1
141 1275 1
142 1276 1
143 1277 1
144 1278 1
145 1279 1
146 1280 1
147 1281 1
148 1282 1

TABLE OF CONTENTS:

FORWARD ROUTINE
    OUTPUT      ; output a counted string to the SCRPKG
    POSITION     ; call SCRPKG to position cursor
    TEMPLATE    ; build and output display templates

INCLUDE FILES:

LIBRARY 'SYSS$LIBRARY:LIB.L32'; ; system service macros and user definitions
REQUIRE 'MONDEFREQ'; ; private MONITOR control block definitions
REQUIRE 'DSPDEFREQ'; ; item numbers defined here

BUILTIN EMUL ; ; define EMUL VAX hardware function

COMPILE TIME VARIABLES

COMPILETIME
    RWAIT_COUNT = 0 ; counter for the number of RWAITs being defined
    RWAIT_DEFINED = RSNS_MAX ; number of RSNS_* wait codes defined in LIB.L32

MACROS:

MACRO

    Counted ascii string macros

CSTRING[] = (UPLIT BYTE(%CHARCOUNT(%STRING(%REMAINING))),
              %STRING(%REMAINING)) % ,

    The RWAIT_CSTRING macro is the CSTRING macro plus a counter to
    keep track of times it was called (how many RWAITS have been defined)

RWAIT_CSTRING[] = %ASSIGN(RWAIT_COUNT,RWAIT_COUNT+1)
                  (UPLIT BYTE(%CHARCOUNT(%STRING(%REMAINING))),
                   %STRING(%REMAINING)) % ;

EQUATED SYMBOLS:

LITERAL

    BELL = 7 ;
    ESC = 27 ;

```



```

149 1283 1 ALTSET = ('F' ^ 8) + ESC,      ! alternate graphics set
150 1284 1 CR = 13,                    ! carriage return
151 1285 1 CURSOR = ('Y' ^ 8) + ESC,   ! position cursor command
152 1286 1 ERASE = ('J' ^ 8) + ESC,    ! erase entire screen
153 1287 1 ERASEEOL = ('K' ^ 8) + ESC, ! erase to end of line
154 1288 1 FALSE = 0,
155 1289 1 HOME = ('H' ^ 8) + ESC,     ! return cursor to top
156 1290 1 LF = 10,                    ! line feed
157 1291 1 TRUE = 1;
158 1292 1
159 1293 1 GLOBAL LITERAL
160 1294 1
161 1295 1 REGSET = ('G' ^ 8) + ESC;    ! normal graphics set
162 1296 1
163 1297 1
164 1298 1
165 1299 1 ! OWN STORAGE:
166 1300 1
167 1301 1
168 1302 1 OWN
169 1303 1 TOPSTR10: VECTOR[45,BYTE]
170 1304 1 INITIAL (BYTE(44),BYTE(' [!3OW,!3OW] !16AC!AC!5<!#UL!>!AC'),
171 1305 1 BYTE(ESC),BYTE('F!#*a'),BYTE(ESC),BYTE('G'),BYTE(ESC),BYTE('K')) ;
172 1306 1
173 1307 1
174 1308 1 ! Table of bit vectors which "illustrate" the pattern of data line
175 1309 1 ! spacing within the data portion of the display screen. There is
176 1310 1 ! one bit vector for each possible number of data items (24). Each
177 1311 1 ! bit vector contains 24 bits representing the lines in the data
178 1312 1 ! portion of the display screen. A '1' bit means this is a data line;
179 1313 1 ! a '0' bit means this is a space. The bits read from right to left;
180 1314 1 ! so, for example, the bit representing line 1 is the right-most.
181 1315 1
182 1316 1
183 1317 1
184 1318 1 OWN
185 1319 1 SCR_PATTERN: VECTOR[24,LONG] INITIAL (
186 1320 1
187 1321 1 LONG(XB'000000000100000000000000'), ! 1 data item
188 1322 1 LONG(XB'000000001010000000000000'), ! 2 data items
189 1323 1 LONG(XB'000000100100100000000000'), ! 3 data items
190 1324 1 LONG(XB'000000101010100000000000'), ! 4 data items
191 1325 1 LONG(XB'000001010101010000000000'), ! 5 data items
192 1326 1 LONG(XB'000010100101001010000000'), ! 6 data items
193 1327 1 LONG(XB'000010101010101010000000'), ! 7 data items
194 1328 1 LONG(XB'00101010101010101010000000'), ! 8 data items
195 1329 1 LONG(XB'000011100111001110000000'), ! 9 data items
196 1330 1 LONG(XB'00011011011011011010000000'), ! 10 data items
197 1331 1 LONG(XB'0001101110111011101100000000'), ! 11 data items
198 1332 1 LONG(XB'0011101110111011101110000000'), ! 12 data items
199 1333 1 LONG(XB'00111101111101111110000000'), ! 13 data items
200 1334 1 LONG(XB'00111111101111111110000000'), ! 14 data items
201 1335 1 LONG(XB'00111111111111111110000000'), ! 15 data items
202 1336 1 LONG(0), ! 16 data items
203 1337 1 LONG(0), ! 17 data items
204 1338 1 LONG(0), ! 18 data items
205 1339 1 LONG(0), ! 19 data items

```

```

206 1340 1 LONG(0), ; 20 data items
207 1341 1 LONG(0), ; 21 data items
208 1342 1 LONG(0), ; 22 data items
209 1343 1 LONG(0), ; 23 data items
210 1344 1 LONG(0); ; 24 data items
211 1345 1
212 1346 1
213 1347 1
214 1348 1 One of the above longword elements is moved to the 24-bit vector
215 1349 1 below, based on the number of items in the display. The bit vector
216 1350 1 is then used to determine whether a line in the data portion of the
217 1351 1 screen is to be a space (0) or is to contain data (1).
218 1352 1
219 1353 1 OWN
220 1354 1 SCR_DATA_LINE: BITVECTOR[24];
221 1355 1
222 1356 1
223 1357 1 Messages
224 1358 1
225 1359 1
226 1360 1 BIND
227 1361 1
228 1362 1 TABSTR = CSTRING(' !7UL !2ZL !7UL !2ZL !7UL !2ZL !7UL !2ZL'),
229 1363 1 TABSTR_PC = CSTRING(' !7UL !1ZL !7UL !1ZL !7UL !1ZL !7UL !1ZL'),
230 1364 1 COUNTSTR = UPLIT BYTE ('!7<!#UL!>'),
231 1365 1 CRSTR = CSTRING(%CHAR(CR)),
232 1366 1 CLRSTR = CSTRING(%CHAR(ESC), 'H', %CHAR(ESC), 'J'),
233 1367 1 DELSTR = CSTRING(%CHAR(ESC), 'J'),
234 1368 1 GRAPHICS_ON = CSTRING(%CHAR(ESC), '1'),
235 1369 1 GRAPHICS_OFF = CSTRING(%CHAR(ESC), '2'),
236 1370 1 HOMESTR = CSTRING(%CHAR(ESC), 'H'),
237 1371 1
238 1372 1 LFSTR = CSTRING(%CHAR(LF)),
239 1373 1 NLSTR = CSTRING(%CHAR(CR), %CHAR(LF)),
240 1374 1 REPTSTR = UPLIT BYTE('!#*'),
241 P 1375 1 SETVT55 = CSTRING(%CHAR(ESC), '1', 'A', %CHAR(%'77'), 'I',
242 1376 1 %CHAR(%'57'), %CHAR(ESC), '2'),
243 1377 1 TOPSTR20 = CSTRING(%CHAR(ESC), 'K'),
244 1378 1 VHSTSTR20 = CSTRING('!UL');
245 1379 1
246 1380 1
247 1381 1 Table of counted strings for Process States
248 1382 1
249 1383 1
250 1384 1 GLOBAL BIND
251 1385 1
252 1386 1 STATELIST = UPLIT ( CSTRING('BAD') ,
253 1387 1 CSTRING('COLPG') ,
254 1388 1 CSTRING('MWAIT') ,
255 1389 1 CSTRING('CEF') ,
256 1390 1 CSTRING('PFW') ,
257 1391 1 CSTRING('LEF') ,
258 1392 1 CSTRING('LEFO') ,
259 1393 1 CSTRING('HIB') ,
260 1394 1 CSTRING('HIBO') ,
261 1395 1 CSTRING('SUSP') ,
262 1396 1 CSTRING('SUSPO') ,

```



```

: 263      1397 1      CSTRING('FPG')      ;
: 264      1398 1      CSTRING('COM')      ;
: 265      1399 1      CSTRING('COMO')     ;
: 266      1400 2      CSTRING('CUR')     ;
: 267      1401 1      ),
: 268      1402 1
: 269      1403 1  RWAITLIST = UPLIT (
: 270      1404 1      RWAIT_CSTRING('RWUDF') ,
: 271      1405 1      RWAIT_CSTRING('RWAST') ,
: 272      1406 1      RWAIT_CSTRING('RWMBX') ,
: 273      1407 1      RWAIT_CSTRING('RWMPG') ,
: 274      1408 1      RWAIT_CSTRING('RWPAG') ,
: 275      1409 1      RWAIT_CSTRING('RWBRK') ,
: 276      1410 1      RWAIT_CSTRING('RWIMG') ,
: 277      1411 1      RWAIT_CSTRING('RWQUO') ,
: 278      1412 1      RWAIT_CSTRING('RWLCK') ,
: 279      1413 1      PWAIT_CSTRING('RWSWP') ,
: 280      1414 1      RWAIT_CSTRING('RWMPE') ,
: 281      1415 1      RWAIT_CSTRING('RWMPB') ,
: 282      1416 1      RWAIT_CSTRING('RWSCS') ,
: 283      1417 2      RWAIT_CSTRING('RWCLU') ,
: 284      1418 1
: 285      1419 1
: 286      1420 1  ! Make sure MONITOR knows all RSN$_* wait states currently defined in LIB.L32
: 287      1421 1
: 288      1422 1      $ASSUME (RWAIT_COUNT, EQL, RWAIT_DEFINED)
: 289      1423 1
: 290      1424 2  RWAITLIST = UPLIT (      CSTRING('MUTEX')
: 291      1425 1      );
: 292      1426 1

```



```

294 1427 1 1
295 1428 1 1
296 1429 1 1
297 1430 1 1
298 1431 1 1
299 1432 1 1
300 1433 1 1
301 1434 1 1
302 1435 1 1
303 1436 1 1
304 1437 1 1
305 1438 1 1
306 1439 1 1
307 1440 1 1
308 1441 1 1
309 1442 1 1
310 1443 1 1
311 1444 1 1
312 1445 1 1
313 1446 1 1
314 1447 1 1
315 1448 1 1
316 1449 1 1
317 1450 1 1
318 1451 1 1
319 1452 1 1
320 1453 1 1
321 1454 1 1
322 1455 1 1
323 1456 1 1
324 1457 1 1
325 1458 1 1
326 1459 1 1
327 1460 1 1
328 1461 1 1
329 1462 1 1
330 1463 1 1
331 1464 1 1
332 1465 1 1
333 1466 1 1

EXTERNAL REFERENCES:

EXTERNAL
MRBPTR , address of MRB
NAME_COL: BYTE , column number for name string
BARCHAR: BYTE , character to repeat to form bar graphs
DISPLAYING: BYTE , low bit set => display is active
FAOSTK: VECTOR[ ,LONG] , fao parameter space
MFSUMSTR , fao string segment for control string
NAMESTR , fao string for output of long names
NORMAL , MONITOR normal return status
PERFTABLE: VECTOR[ ,BYTE] , ! list of performance item descriptors
ITMSTR_SYS_ALL: BYTE , ! item string for SYSTEM /ALL
SCH$GL_MAXPIX: ADDRESSING_MODE(LONG_RELATIVE) , ! max process index
SCH$GL_PCBVEC: ADDRESSING_MODE(LONG_RELATIVE) , ! address of PCB pointer list
VT$SXINCR ; ! incr from bar to bar

EXTERNAL LITERAL
FAOCTR_SIZE , size of FAO control string
FIRST_DATA_LINE , line number of first data line on screen
LAST_DATA_LINE , line number of last data line on screen
VTDATALINES , number of data lines on the screen
NAME_COL_TAB , starting column of names -- tabular display
NAME_COL_BAR , starting column of names -- bar graph
NAME_COL_MFSUM , starting column of names -- multi-file summary
MAX_NAME_SIZE , max size of name (label) string
WIDE_NAME_SIZE , size of name (label) string for a wide display (DISK)
ECOUNT_SYS_ALL , no. of elements for SYSTEM /ALL
MAXBARS , max characters on horizontal histogram
VT$SCWIDTH , max characters on bottom axis
VTHEIGHT , height of screen
VTWIDTH ; width of screen

EXTERNAL ROUTINE
PUT_TO_SCREEN , ! rtn to xlate & annex a string to SYSS$OUTPUT buffer
LIB$GET_VM , ! rtn to acquire virtual memory
SCR$SET_CURSOR ; ! rtn to annex a cursor positioning esc seq to SYSS$OUTPUT

```

```

335 1467 1 GLOBAL ROUTINE TEMPLATE( DCDB ) =
336 1468 BEGIN
337 1469
338 1470 --
339 1471
340 1472 FUNCTIONAL DESCRIPTION:
341 1473
342 1474 This routine formats and displays the name strings for tabular
343 1475 and bar graph displays of current, average, min and max values.
344 1476 It also builds the FAO control string for the actual data on the
345 1477 first call per class.
346 1478
347 1479 INPUTS:
348 1480
349 1481 DCDB - address of class descriptor block for class being displayed.
350 1482
351 1483 IMPLICIT INPUTS:
352 1484
353 1485 PERFTable - address of table of contiguous IDB's.
354 1486
355 1487
356 1488 OUTPUTS:
357 1489
358 1490 none
359 1491
360 1492 IMPLICIT OUTPUTS:
361 1493
362 1494 Name string for each item in the display for this class sent
363 1495 directly to screen package (via call to PUT_TO_SCREEN).
364 1496
365 1497 On first call to this routine for this class, a buffer is
366 1498 obtained for the FAO control string to output the data values.
367 1499 It is filled with the necessary FAO control information and
368 1500 its address and length are stored in the CDB$A_FAOCTR and
369 1501 CDB$S_FAOCTR fields, respectively.
370 1502
371 1503 ROUTINE VALUE:
372 1504
373 1505 NORMAL, or possible failing status from LIB$GET_VM.
374 1506
375 1507 SIDE EFFECTS:
376 1508
377 1509 none
378 1510 --
379 1511
380 1512 LOCAL
381 1513
382 1514 I,
383 1515 ITEMS,
384 1516 ITMSTR,
385 1517 POINTER,
386 1518 STATUS,
387 1519 XPOS,
388 1520 YPOS,
389 1521 ROW_OFFSET;
390 1522
391 1523 MAP
392 1524 DCDB: REF BLOCK[,BYTE] ;
393 1525 MRBPTR: REF BLOCK[,BYTE] ;
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000

```

```

| data item index
| count of data items
| pointer to first item token
| pointer into fao control string buffer
| return status
| column address
| row address
| constant added to row number for m.f. summary
|
| address CDB structure
| address MRB structure

```

TEMPLATE
V04-000

N 4
16-Sep-1984 02:18:37 YAX-11 Bliss-32 V4.0-742
14-Sep-1984 12:45:05 [MONITOR.SRC]TEMPLATE.B32;1

Page 9
(4)

; 392

1524 2

ITMSTR: REF VECTOR[.BYTE] ;

! item byte string


```

394 1525 2 IF .MRBPTR[MRBSV MFSUM]           ! if this is a multi-file summary
395 1526      THEN ROW_OFFSET = 2           ! then display the data rows lower
396 1527      ELSE ROW_OFFSET = 0 ;         ! else do not offset
397 1528
398 1529 IF .DCDB[DCBSV HOMOG]               ! if this is a homogeneous class,
399 1530      THEN ITEMS = VTDATALINES         ! always use the whole screen,
400 1531      ELSE ITEMS = .DCDB[DCBSL_ECOUN] ! else get just no. of elts to display
401 1532
402 1533 IF .DCDB[DCBSV SYSCLS]                 ! if this is the SYSTEM class,
403 1534      THEN ITEMS = ECOUNT_SYS_ALL ;    ! get a special ECOUNT
404 1535
405 1536 SCR TA_LINE = 0;                     ! zero out display bit string
406 1537
407 1538
408 1539 ! Set up bit string controlling spacing.
409 1540 ! The CDB display control string is only a word in length, rather than 24 bits.
410 1541 ! This is to save space, since only 15 of the 24 bits in the default bit
411 1542 ! strings are actually used.
412 1543
413 1544
414 1545 IF .DCDB[DCBSW DISPCTL] EQL 0           ! if display control is 0
415 1546      THEN SCR_DATA_LINE = .(SCR_PATTE[ITEMS-1])<0,24> ! use default spacing
416 1547      ELSE SCR_DATA_LINE<7,15> = .(DCDB[DCBSW_DISPCTL])<0,15> ; ! else use spacing specified in CDB
417 1548
418 1549 ! Output name string for each item in this heterogeneous class
419 1550
420 1551
421 1552 IF .MRBPTR[MRBSV MFSUM] OR .DCDB[DCBSV_WIDE] ! if this is a multi-file summary or a wide screen
422 1553      THEN NAME_COL = NAME_COL_MFSUM      ! start the names here
423 1554      ELSE IF .DCDB[DCBSB_ST] EQL ALL_STAT ! if this is a tabular display,
424 1555          THEN NAME_COL = NAME_COL_TAB    ! start the names here
425 1556          ELSE NAME_COL = NAME_COL_BAR ;   ! else start there for bar graph
426 1557
427 1558 IF NOT .DCDB[DCBSV_HOMOG]                ! if this is a heterogeneous class,
428 1559 THEN
429 1560     BEGIN
430 1561
431 1562     I = 0 ;                               ! initialize data item index
432 1563     ITMSTR = .DCDB[DCBSA_ITMSTR] ;         ! get address of item byte string
433 1564
434 1565     IF .DCDB[DCBSV_SYSCLS] AND .DCDB[DCBSB_ST] EQL ALL_STAT ! if this is the SYSTEM tabular display,
435 1566         THEN ITMSTR = ITMSTR_SYS_ALL ;    ! get a special ITMSTR
436 1567
437 1568     INCR YPOS FROM FIRST_DATA_LINE TO LAST_DATA_LINE ! loop once for each line in
438 1569     DO                                              ! ... data portion of screen
439 1570     BEGIN
440 1571
441 1572     !
442 1573     ! Find the IDB for this item. Output the long name
443 1574     ! string, preceded by the correct cursor positioning
444 1575     ! sequence to space them out evenly.
445 1576     !
446 1577
447 1578     LOCAL
448 1579     DIDB: REF BLOCK[.BYTE] ,
449 1580     NAME ;
450 1581     NEXT ;

```

| | | | | |
|-----|------|---|---|---------------------------------|
| 451 | 1582 | 4 | IF .SCR_DATA_LINE[.YPOS-1] | ! if this is a data line, |
| 452 | 1583 | 4 | THEN | |
| 453 | 1584 | 5 | BEGIN | |
| 454 | 1585 | 5 | NEXT = .ITMSTR[.I] ; | ! get next token |
| 455 | 1586 | 5 | DIDB = PERFTABLE[.NEXT * IDBSK_ILENGTH] ; | ! addr of IDB |
| 456 | 1587 | 5 | NAME = .DIDB[IDBSA_LNAME] ; | ! address of name string |
| 457 | 1588 | 5 | POSITION(.YPOS + .ROW_OFFSET , .NAME_COL) ; | ! position to this item |
| 458 | 1589 | 5 | OUTPUT(NAME) ; | ! output name string |
| 459 | 1590 | 5 | IF .DIDB[IDBSV_PCNT] EQL 1 | ! if this is a pcnt item |
| 460 | 1591 | 5 | THEN I = .I + 2 | ! move past item used for calc |
| 461 | 1592 | 5 | ELSE I = .I + 1; | ! point index to next data item |
| 462 | 1593 | 4 | END; | |
| 463 | 1594 | 4 | | |
| 464 | 1595 | 3 | END; | |
| 465 | 1596 | 2 | END; | |

```

467 1597 2 |
468 1598 2 |
469 1599 2 | If this is the first time thru for this class,
470 1600 2 | obtain and build the FAO control string to insert
471 1601 2 | the data values for the items at data display time.
472 1602 2 |
473 1603 2 | IF .DCDB[CDB$A_FAOCTR] EQL 0 OR NOT .DISPLAYING | if no fao control string yet
474 1604 2 | THEN | ... OR in summary processing
475 1605 2 | BEGIN
476 1606 2 | LOCAL
477 1607 2 | FAOCSIZE ; | holds faoctr size
478 1608 2 | IF .DCDB[CDB$A_FAOCTR] EQL 0 | if no control string buffer yet,
479 1609 2 | THEN
480 1610 2 | BEGIN
481 1611 2 | FAOCSIZE = FAOCTR SIZE ; | initialize its size
482 1612 2 | STATUS = LIB$GET_VM(FAOCSIZE,DCDB[CDB$A_FAOCTR]); | get the memory for it
483 1613 2 | IF NOT .STATUS THEN RETURN .STATUS ; | return if error
484 1614 2 | END;
485 1615 2 |
486 1616 2 | POINTER = .DCDB[CDB$A_FAOCTR] ; | start pointer at beg of FAO buffer
487 1617 2 |
488 1618 2 | IF .DCDB[CDB$B_ST] EQL ALL_STAT OR .MRBPTR[MRB$V_MFSUM] | if this is a tabular display,
489 1619 2 | THEN | set up control string accordingly
490 1620 2 | BEGIN
491 1621 2 | LOCAL
492 1622 2 | COL_OFFSET ; | holds offset from usual column where data
493 1623 2 | CUR_TABSTR ; | holds addr of FAO control string segment
494 1624 2 | IF .DCDB[CDB$V_WIDE] | if a wide-screen display,
495 1625 2 | THEN COL_OFFSET = WIDE_NAME_SIZE | then set a wide offset
496 1626 2 | ELSE COL_OFFSET = MAX_NAME_SIZE ; | otherwise, take the usual width
497 1627 2 | XPOS = .NAME_COL + .COL_OFFSET ; | starting column
498 1628 2 | DCDB[CDB$B_FAOPRELEN] = 0 ; | length of FAO prefix
499 1629 2 |
500 1630 2 | IF .MRBPTR[MRB$V_MFSUM] | if this is a multi-file summary,
501 1631 2 | THEN CUR_TABSTR = MFSUMSTR | get the appropriate FAO control str segm
502 1632 2 | ELSE IF .DCDB[CDB$V_PERCENT] | if this is a percent display,
503 1633 2 | THEN CUR_TABSTR = TABSTR_PC | get the appropriate FAO control str segm
504 1634 2 | ELSE CUR_TABSTR = TABSTR ; | else get the other one
505 1635 2 |
506 1636 2 | INCR YPOS FROM FIRST_DATA_LINE TO LAST_DATA_LINE | loop once for each line in
507 1637 2 | DO | ... data portion of screen
508 1638 2 | BEGIN
509 1639 2 | IF .SCR_DATA_LINE[.YPOS-1] | if this is a data line,
510 1640 2 | THEN
511 1641 2 | BEGIN
512 1642 2 | (.POINTER)<0,16> = CURSOR ; | insert position command
513 1643 2 | (POINTER = .POINTER + 2)<0,8> = .YPOS + .ROW_OFFSET ; | insert row number
514 1644 2 | (POINTER = .POINTER + 1)<0,8> = .XPOS ; | insert column number
515 1645 2 | POINTER = .POINTER + 1 ; | update to skip last inserted byte
516 1646 2 | CH$MOVE(.(CUR_TABSTR)<0,8>,(CUR_TABSTR)+1,.POINTER) ; | move conversion control stri
517 1647 2 | POINTER = .POINTER + .(CUR_TABSTR)<0,8> ; | update pointer
518 1648 2 | IF .YPOS EQL FIRST_DATA_LINE | if first time thru the loop,
519 1649 2 | THEN DCDB[CDB$B_FAOSEGLN] = .POINTER - .DCDB[CDB$A_FAOCTR] - .DCDB[CDB$B_FAOPRELEN] ; | compute length of a single segment
520 1650 2 |
521 1651 2 | END;
522 1652 2 |
523 1653 2 | END;

```


TEMPLATE
V04-000

: 524

1654 4

END

E 5
16-Sep-1984 02:18:37
14-Sep-1984 12:45:05

VAX-11 Bliss-32 V4.0-742
[MONITOR.SRC]TEMPLATE.B32;1

Page 13
(6)

EXI

MO
--
MO
CO
CH
GE
CH
CH
IN
SN
DI
CL
LO
CH
GE
LE
RE
AL
VM
RD
ST
MO
MO
MO
MR
IN
SR
TR
HU
HU
MA
BI
CH
CH
MA
ER
CL
AS
MO
RU
SY
CJ
SY
LI
LI
LI
LI
LI
LI

```

526 1655 3 ELSE
527 1656 4 BEGIN
528 1657 4
529 1658 4 Now build the fao control string to output a bar graph
530 1659 4 at run time. The control string contains for each line:
531 1660 4 position row and column to left of grid
532 1661 4 write count
533 1662 4 re-position row and column inside grid
534 1663 4 output 'n' bar characters
535 1664 4 delete to end of line
536 1665 4
537 1666 4 LOCAL
538 1667 4 XPOSBAR ! column number of beg of bar
539 1668 4 XPOSCOUNT ; ! column number of count field
540 1669 4
541 1670 4 XPOSCOUNT = 30 ; ! starting column of count field
542 1671 4 XPOSBAR = 39 ; ! starting column of bar field
543 1672 4 (.POINTER) <0,16> = ALTSET ; ! start filling ctrl string (alternate graphics)
544 1673 4 POINTER = .POINTER + 2 ; ! skip to next position
545 1674 4 DCDB[CDB$B_FAOPRELEN] = 2 ; ! ... and store length of FAO prefix
546 1675 4
547 1676 4 INCR YPOS FROM FIRST_DATA_LINE TO LAST_DATA_LINE ! loop once for each line in
548 1677 4 DO ! ... data portion of screen
549 1678 5 BEGIN
550 1679 5 IF .SCR_DATA_LINE[YPOS-1] ! if this is a data line,
551 1680 5 THEN
552 1681 6 BEGIN
553 1682 6 (.POINTER)<0,16> = CURSOR ; ! insert position command
554 1683 6 (POINTER = .POINTER + 2 )<0,8> = .YPOS ; ! next Y position
555 1684 6 (POINTER = .POINTER + 1 )<0,8> = .XPOSCOUNT ; ! X position for count
556 1685 6 POINTER = .POINTER + 1 ; ! next buffer position
557 1686 6 CHSMOVE( 9 , COUNTSTR , .POINTER ) ; ! move count directive
558 1687 6 (POINTER = .POINTER+9)<0,16> = CURSOR ; ! insert control to position to
559 1688 6 (POINTER = .POINTER+2)<0,8> = .YPOS ; ! stay in same row
560 1689 6 (POINTER = .POINTER+1)<0,8> = .XPOSBAR ; ! column for bar field
561 1690 6 POINTER = .POINTER + 1 ; ! next buffer position
562 1691 6 CHSMOVE( 3 , REPTSTR , .POINTER ) ; ! move repeat control
563 1692 6 (POINTER = .POINTER + 3)<0,8> = .BARCHAR ; ! insert literal character to use for graph
564 1693 6 (POINTER = .POINTER+1)<0,16> = ERASEOL ; ! delete rest of line
565 1694 6 POINTER = .POINTER + 2 ; ! next buffer position
566 1695 6 IF .YPOS EQL FIRST_DATA_LINE ! if first time thru the loop,
567 1696 6 THEN DCDB[CDB$B_FAOSEGLN] = .POINTER - .DCDB[CDB$A_FAOCTR] - .DCDB[CDB$B_FAOPRELEN] ; ! compute length of a single segment
568 1697 6
569 1698 5 END;
570 1699 4 END;
571 1700 4
572 1701 4 (.POINTER)<0,16> = REGSET ; ! restore normal char set
573 1702 4 POINTER = .POINTER + 2 ; ! update position
574 1703 4 END;
575 1704 4
576 1705 4 !
577 1706 4 ! Insert length of created string into CDB
578 1707 4 !
579 1708 4
580 1709 3 DCDB[CDB$L_FAOCTR] = .POINTER - .DCDB[CDB$A_FAOCTR] ;
581 1710 2 END ;
582 1711 2 RETURN .NORMAL ; ! return with no errors

```

| | | | | | | | | | | | | | | | | | | | | |
|----|----|----|----|----|----|----|----|----|----|----|----|----|----|--------|------------------------|--------|--------|---|---------|--------|
| | | | | | | | | | | | | | | .TITLE | TEMPLATE | | | | | |
| | | | | | | | | | | | | | | .IDENT | \V04-000\ | | | | | |
| | | | | | | | | | | | | | | .PSECT | \$SPLITS,NOWRT,NOEXE,2 | | | | | |
| 55 | 37 | 21 | 20 | 4C | 5A | 32 | 21 | 2E | 4C | 55 | 37 | 21 | 20 | 29 | 00000 | P.AAA: | .BYTE | 41 | | |
| 5A | 32 | 21 | 2E | 4C | 55 | 37 | 21 | 20 | 4C | 5A | 32 | 21 | 2E | 20 | 00001 | | .ASCII | \ !7UL.!2ZL !7UL.!2ZL !7UL.!2ZL !7UL.!2Z\ | | |
| | | | | | 5A | 32 | 21 | 2E | 4C | 55 | 37 | 21 | 20 | 4C | 00010 | | | | | |
| | | | | | | | | | | | | | | 4C | 0001F | | | | | |
| | | | | | | | | | | | | | | 20 | 00029 | P.AAB: | .ASCII | \L\ | | |
| 21 | 20 | 20 | 4C | 5A | 31 | 21 | 2E | 4C | 55 | 37 | 21 | 20 | 20 | 20 | 0002A | | .BYTE | 45 | | |
| 2E | 4C | 55 | 37 | 21 | 20 | 20 | 4C | 5A | 31 | 21 | 2E | 4C | 55 | 20 | 0002B | | .ASCII | \ !7UL.!1ZL !7UL.!1ZL !7UL.!1ZL !7UL\ | | |
| | | | | | 4C | 55 | 37 | 21 | 20 | 20 | 4C | 5A | 31 | 21 | 0003A | | | | | |
| | | | | | | | | | | | | | | 2E | 00049 | | | | | |
| | | | | | | | | | | | | | | 21 | 00053 | | .ASCII | \.!1ZL\ | | |
| | | | | | 3E | 21 | 4C | 55 | 23 | 21 | 3C | 37 | | 01 | 00058 | P.AAC: | .ASCII | \!7<!#UL!>\ | | |
| | | | | | | | | | | | | | | 0D | 00061 | P.AAD: | .BYTE | 1 | | |
| | | | | | | | | | | | | | | 04 | 00062 | | .ASCII | <13> | | |
| | | | | | | | | | | | | | | 04 | 00063 | P.AAE: | .BYTE | 4 | | |
| | | | | | | | | | | 4A | 1B | 48 | 1B | 02 | 00064 | | .ASCII | <27>\H\<27>\J\ | | |
| | | | | | | | | | | | | | | 02 | 00068 | P.AAF: | .BYTE | 2 | | |
| | | | | | | | | | | | | | 4A | 1B | 00069 | | .ASCII | <27>\J\ | | |
| | | | | | | | | | | | | | | 02 | 0006B | P.AAG: | .BYTE | 2 | | |
| | | | | | | | | | | | | | 31 | 1B | 0006C | | .ASCII | <27>\1\ | | |
| | | | | | | | | | | | | | | 02 | 0006E | P.AAH: | .BYTE | 2 | | |
| | | | | | | | | | | | | | 32 | 1B | 0006F | | .ASCII | <27>\2\ | | |
| | | | | | | | | | | | | | | 02 | 00071 | P.AAI: | .BYTE | 2 | | |
| | | | | | | | | | | | | | 48 | 1B | 00072 | | .ASCII | <27>\H\ | | |
| | | | | | | | | | | | | | | 01 | 00074 | P.AAJ: | .BYTE | 1 | | |
| | | | | | | | | | | | | | | 0A | 00075 | | .ASCII | <10> | | |
| | | | | | | | | | | | | | | 02 | 00076 | P.AAK: | .BYTE | 2 | | |
| | | | | | | | | | | | | | 0A | 0D | 00077 | | .ASCII | <13><10> | | |
| | | | | | | | | | | | | | 2A | 23 | 21 | 00079 | P.AAL: | .ASCII | \!#*\ | |
| | | | | | | | | | | | | | | 08 | 0007C | P.AAM: | .BYTE | 8 | | |
| | | | | | | | | | | | | | | 08 | 0007D | | .ASCII | <27>\1A?I/\<27>\2\ | | |
| | | | | | | | | | | | | | | 02 | 00085 | P.AAN: | .BYTE | 2 | | |
| | | | | | | | | | | | | | | 4B | 1B | 00086 | | .ASCII | <27>\K\ | |
| | | | | | | | | | | | | | | 03 | 00088 | P.AAO: | .BYTE | 3 | | |
| | | | | | | | | | | | | | 4C | 55 | 21 | 00089 | | .ASCII | \!UL\ | |
| | | | | | | | | | | | | | | 03 | 0008C | P.AAQ: | .BYTE | 3 | | |
| | | | | | | | | | | | | | 44 | 41 | 42 | 0008D | | .ASCII | \BAD\ | |
| | | | | | | | | | | | | | | 05 | 00090 | P.AAR: | .BYTE | 5 | | |
| | | | | | | | | | | | | | 47 | 50 | 4C | 4F | 43 | 00091 | | |
| | | | | | | | | | | | | | | 05 | 00096 | P.AAS: | .BYTE | 5 | | |
| | | | | | | | | | | | | | 54 | 49 | 41 | 57 | 4D | 00097 | | |
| | | | | | | | | | | | | | | 03 | 0009C | P.AAT: | .BYTE | 3 | | |
| | | | | | | | | | | | | | | 46 | 45 | 43 | 0009D | | .ASCII | \CEF\ |
| | | | | | | | | | | | | | | 03 | 000A0 | P.AAU: | .BYTE | 3 | | |
| | | | | | | | | | | | | | 57 | 46 | 50 | 000A1 | | .ASCII | \PFW\ | |
| | | | | | | | | | | | | | | 03 | 000A4 | P.AAV: | .BYTE | 3 | | |
| | | | | | | | | | | | | | 46 | 45 | 4C | 000A5 | | .ASCII | \LEF\ | |
| | | | | | | | | | | | | | | 04 | 000A8 | P.AAW: | .BYTE | 4 | | |
| | | | | | | | | | | | | | 4F | 46 | 45 | 4C | 000A9 | | .ASCII | \LEFO\ |
| | | | | | | | | | | | | | | 03 | 000AD | P.AAX: | .BYTE | 3 | | |

| | | | | | | | | | | | |
|-----------|-----------|-----------|-----------|-----------|-----------|-----------|-------|--------|----------|---|--|
| | | | | 42 | 49 | 48 | 000AE | | .ASCII | \HIB\ | |
| | | | | | | 04 | 000B1 | P.AAY: | .BYTE | 4 | |
| | | | 4F | 42 | 49 | 48 | 000B2 | | .ASCII | \HIB0\ | |
| | | | | | | 04 | 000B6 | P.AAZ: | .BYTE | 4 | |
| | | | 50 | 53 | 55 | 53 | 000B7 | | .ASCII | \SUSP\ | |
| | | | | | | 05 | 000BB | P.ABA: | .BYTE | 5 | |
| | | | 4F | 50 | 53 | 55 | 000BC | | .ASCII | \SUSPO\ | |
| | | | | | | 03 | 000C1 | P.ABB: | .BYTE | 3 | |
| | | | | 47 | 50 | 46 | 000C2 | | .ASCII | \FPG\ | |
| | | | | | | 03 | 000C3 | P.ABC: | .BYTE | 3 | |
| | | | | 4D | 4F | 43 | 000C6 | | .ASCII | \COM\ | |
| | | | | | | 04 | 000C9 | P.ABD: | .BYTE | 4 | |
| | | | 4F | 4D | 4F | 43 | 000CA | | .ASCII | \COM0\ | |
| | | | | | | 03 | 000CE | P.ABE: | .BYTE | 3 | |
| | | | | 52 | 55 | 43 | 000CF | | .ASCII | \CUR\ | |
| | | | | | | | 000D2 | | .BLKB | 2 | |
| 00000000' | 00000000' | 00000000' | 00000000' | 00000000' | 00000000' | 00000000' | 000D4 | P.AAP: | .ADDRESS | P.AAQ, P.AAR, P.AAS, P.AAT, P.AAU, - | |
| 00000000' | 00000000' | 00000000' | 00000000' | 00000000' | 00000000' | 00000000' | 000EC | | | P.AAV, P.AAW, P.AAX, P.AAY, P.AAZ, P.ABA, - | |
| | | | | | | 00000000' | 00104 | | | P.ABB, P.ABC, P.ABD, P.ABE | |
| | | | | | | 05 | 00110 | P.ABG: | .BYTE | 5 | |
| | | | 46 | 44 | 55 | 57 | 00111 | | .ASCII | \RWUDF\ | |
| | | | | | | 05 | 00116 | P.ABH: | .BYTE | 5 | |
| | | | 54 | 53 | 41 | 57 | 00117 | | .ASCII | \RWAST\ | |
| | | | | | | 05 | 0011C | P.ABI: | .BYTE | 5 | |
| | | | 58 | 42 | 4D | 57 | 0011D | | .ASCII | \RWMBX\ | |
| | | | | | | 05 | 00122 | P.ABJ: | .BYTE | 5 | |
| | | | 47 | 50 | 4E | 57 | 00123 | | .ASCII | \RWNPG\ | |
| | | | | | | 05 | 00128 | P.ABK: | .BYTE | 5 | |
| | | | 46 | 47 | 50 | 57 | 00129 | | .ASCII | \RWPGF\ | |
| | | | | | | 05 | 0012E | P.ABL: | .BYTE | 5 | |
| | | | 47 | 41 | 50 | 57 | 0012F | | .ASCII | \RWPAG\ | |
| | | | | | | 05 | 00134 | P.ABM: | .BYTE | 5 | |
| | | | 48 | 52 | 42 | 57 | 00135 | | .ASCII | \RWBRK\ | |
| | | | | | | 05 | 0013A | P.ABN: | .BYTE | 5 | |
| | | | 47 | 4D | 49 | 57 | 0013B | | .ASCII | \RWIMG\ | |
| | | | | | | 05 | 00140 | P.ABO: | .BYTE | 5 | |
| | | | 4F | 55 | 51 | 57 | 00141 | | .ASCII | \RWQUO\ | |
| | | | | | | 05 | 00146 | P.ABP: | .BYTE | 5 | |
| | | | 48 | 43 | 4C | 57 | 00147 | | .ASCII | \RWLCK\ | |
| | | | | | | 05 | 0014C | P.ABQ: | .BYTE | 5 | |
| | | | 50 | 57 | 53 | 57 | 0014D | | .ASCII | \RWSWP\ | |
| | | | | | | 05 | 00152 | P.ABR: | .BYTE | 5 | |
| | | | 45 | 50 | 4D | 57 | 00153 | | .ASCII | \RWMPE\ | |
| | | | | | | 05 | 00158 | P.ABS: | .BYTE | 5 | |
| | | | 42 | 50 | 4D | 57 | 00159 | | .ASCII | \RWMPB\ | |
| | | | | | | 05 | 0015E | P.ABT: | .BYTE | 5 | |
| | | | 53 | 43 | 53 | 57 | 0015F | | .ASCII | \RWSCS\ | |
| | | | | | | 05 | 00164 | P.ABU: | .BYTE | 5 | |
| | | | 55 | 4C | 43 | 57 | 00165 | | .ASCII | \RWCLU\ | |
| | | | | | | | 0016A | | .BLKB | 2 | |
| 00000000' | 00000000' | 00000000' | 00000000' | 00000000' | 00000000' | 00000000' | 0016C | P.ABF: | .ADDRESS | P.ABG, P.ABH, P.ABI, P.ABJ, P.ABK, - | |
| 00000000' | 00000000' | 00000000' | 00000000' | 00000000' | 00000000' | 00000000' | 00184 | | | P.ABL, P.ABM, P.ABN, P.ABO, P.ABP, P.ABQ, - | |
| | | | | | | 00000000' | 0019C | | | P.ABR, P.ABS, P.ABT, P.ABU | |
| | | | | | | 05 | 001A8 | P.ABW: | .BYTE | 5 | |
| | | | 58 | 45 | 54 | 55 | 4D | | .ASCII | \MUTEX\ | |
| | | | | | | | 001AE | | .BLKB | 2 | |
| | | | | | | 00000000' | 001B0 | P.ABV: | .ADDRESS | P.ABW | |

```

.PSECT $OWNS,NOEXE,2

2C 00000 TOPSTR10:
21 20 20 5D 57 4F 33 21 2C 57 4F 33 21 5B 20 00001 .BYTE 44
21 4C 55 23 21 3C 35 21 43 41 21 43 41 36 31 00010 .ASCII \ [!3OW,!3OW] !16AC!AC!5<!#UL!>!AC\
43 41 21 3E 0001F
18 00023 .BYTE 27
61 2A 23 21 46 00024 .ASCII \F!#*a\
18 00029 .BYTE 27
47 0002A .ASCII \G\
18 0002B .BYTE 27
4B 0002C .ASCII \K\
0002D .BLKB 3
0004000 00030 SCR_PATTERN:
0000A000 00034 .LONG 16384
00024800 00038 .LONG 40960
0002A800 0003C .LONG 149504
00055400 00040 .LONG 174080
000A5280 00044 .LONG 349184
000AAA80 00048 .LONG 676480
002AAA80 0004C .LONG 699008
000E7380 00050 .LONG 2796160
001B6D80 00054 .LONG 947072
001B8B80 00058 .LONG 1797504
003B8B80 0005C .LONG 1817472
003DF780 00060 .LONG 3914624
003FBF80 00064 .LONG 4061056
003FFF80 00068 .LONG 4177792
00000000 0006C .LONG 4194176
00000000 00070 .LONG 0
00000000 00074 .LONG 0
00000000 00078 .LONG 0
00000000 0007C .LONG 0
00000000 00080 .LONG 0
00000000 00084 .LONG 0
00000000 00088 .LONG 0
00000000 0008C .LONG 0
00090 SCR_DATA_LINE:
.BLKB 3

REGSET== 18203
TABSTR= P.AAA
TABSTR_PC= P.AAB
COUNTSTR= P.AAC
CRSTR= P.AAD
CLRSTR= P.AAE
DELSTR= P.AAF
GRAPHICS_ON= P.AAG
GRAPHICS_OFF= P.AAH
HOMESTR= P.AAI
LFSTR= P.AAJ
NLSTR= P.AAK
REPTSTR= P.AAL
SETVT55= P.AAM

```

TOPSTR20=
VHSTSTR20=
STATLIST==
RWAITLIST==
MWAITLIST==

P.AAN
P.AAO
P.AAP
P.ABF
P.ABV

.EXTRN MRBPTR, NAME_COL
.EXTRN BARCHAR, DISPLAYING
.EXTRN FAOSTK, MFSUMSTR
.EXTRN NAMESTR, NORMAL
.EXTRN PERFTABLE, ITMSTR SYS_ALL
.EXTRN SCH\$GL MAXPIX, SCH\$GL PCBVEC
.EXTRN VT55XINCR, FAOCTR_SIZE
.EXTRN FIRST_DATA_LINE
.EXTRN LAST_DATA_LINE, VTDATALINES
.EXTRN NAME_COL_TAB, NAME_COL_BAR
.EXTRN NAME_COL_MFSUM, MAX_NAME_SIZE
.EXTRN WIDE_NAME_SIZE, ECOUNT_SYS_ALL
.EXTRN MAXBARS, VT55CWIDTH
.EXTRN VTHEIGHT, VTWIDTH
.EXTRN PUT TO SCREEN, LIB\$GET_VM
.EXTRN SCR\$SET_CURSOR

.PSECT \$CODE\$,NOWRT,2

.ENTRY TEMPLATE, Save R2,R3,R4,R5,R6,R7,R8,R9,R10,-; 1467
R11
SUBL2 #8, SP
MOVL MRBPTR, R1 1525
BBC #3, 68(R1), 1\$
MOVL #2, ROW_OFFSET 1526
BRB 2\$
CLRL ROW_OFFSET 1527
MOVL DCDB, R8 1529
BBC #5, 75(R8), 3\$
MOVL #VTDATALINES, ITEMS 1530
BRB 4\$
MOVL 24(R8), ITEMS 1531
BLBC 76(R8), 5\$ 1533
MOVL #ECOUNT_SYS_ALL, ITEMS 1534
INSV #0, #0, #24, SCR_DATA_LINE 1536
TSTW 54(R8) 1545
BNEQ 6\$
INSV SCR_PATTERN-4[ITEMS], #0, #24, -
SCR_DATA_LINE 1546
BRB 7\$
INSV 54(R8), #7, #15, SCR_DATA_LINE 1547
RBS #3, 68(R1), 8\$ 1552
BBC #3, 76(R8), 9\$
MOVB #NAME_COL_MFSUM, NAME_COL 1553
BRB 11\$
TSTB 66(R8) 1554
BNEQ 10\$
MOVB #NAME_COL_TAB, NAME_COL 1555
BRB 11\$
MOVB #NAME_COL_BAR, NAME_COL 1556
BBS #5, 75(R8), 16\$ 1558
CLRL I 1562

OFFC 00000
05 44 5E 00000000G 08 C2 00002
51 00000000G 00 D0 00005
A1 03 E1 0000C
5B 02 D0 00011
02 11 00014
5B D4 00016 1\$:
09 4B 5B 04 AC D0 00018 2\$:
A8 05 E1 0001C
50 00000000G 8F D0 00021
04 11 00028
50 18 A8 D0 0002A 3\$:
07 4C A8 E9 0002E 4\$:
50 00000000G 8F D0 00032
00 00 F0 00039 5\$:
36 A8 B5 00042
10 12 00045
00 00000000'EF40 F0 00047
0A 11 00055
0F 07 36 A8 F0 00057 6\$:
05 44 A1 03 E0 00061 7\$:
0A 4C A8 03 E1 00066
00000000G 00 00G 8F 90 00068 8\$:
17 11 00073
42 A8 95 00075 9\$:
0A 12 00078
00000000G 00 00G 8F 90 0007A
08 11 00082
00000000G 00 00G 8F 90 00084 10\$:
6C 4B A8 05 E0 0008C 11\$:
55 D4 00091

| | | | | | | | | | |
|-----------|-----------|-----------|-----------|----|-------|--------|-----------------------------|-----------------------------|------|
| | 54 | 1C | A8 | D0 | 00093 | MOVL | 28(R8), ITMSTR | 1563 | |
| | 0C | 4C | A8 | E9 | 00097 | BLBC | 76(R8), 12\$ | 1565 | |
| | | 42 | A8 | 95 | 0009B | TSTB | 66(R8) | | |
| | | | 07 | 12 | 0009E | BNEQ | 12\$ | | |
| 52 | 00000000G | | 00 | 9E | 000A0 | MOVAB | ITMSTR SYS_ALL, ITMSTR | 1566 | |
| | 8F | | 01 | C3 | 000A7 | SUBL3 | #1, #FIRST_DATA_LINE, YPOS | 1568 | |
| | | | 44 | 11 | 000AF | BRB | 15\$ | | |
| 38 | 00000000' | FF | A2 | 9E | 000B1 | MOVAB | -1(R2), R0 | 1582 | |
| | EF | | 50 | E1 | 000B5 | BBC | R0, SCR_DATA_LINE, 15\$ | | |
| | 50 | | 6544 | 9A | 000BD | MOVZBL | (I)[ITMSTR],-NEXT | 1585 | |
| | 50 | | 11 | C4 | 000C1 | MULL2 | #17, R0 | 1586 | |
| | 53 | 00000000G | 0040 | 9E | 000C4 | MOVAB | PERFTABLE[R0], DIDB | | |
| | 56 | 04 | A3 | D0 | 000CC | MOVL | 4(DIDB), NAME | 1587 | |
| | 7E | 00000000G | 00 | 9A | 000D0 | MOVZBL | NAME_COL, -(SP) | 1588 | |
| | | | 6B42 | 9F | 000D7 | PUSHAB | (ROW-OFFSET)[YPOS] | | |
| 00000000V | EF | | 02 | FB | 000DA | CALLS | #2, POSITION | | |
| | | | 56 | DD | 000E1 | PUSHL | NAME | 1589 | |
| 00000000V | EF | | 01 | FB | 000E3 | CALLS | #1, OUTPUT | | |
| | 05 | 10 | A3 | E9 | 000EA | BLBC | 16(DIDB), 14\$ | 1590 | |
| | 55 | | 02 | C0 | 000EE | ADDL2 | #2, I | 1591 | |
| | | | 02 | 11 | 000F1 | BRB | 15\$ | | |
| | | | 55 | D6 | 000F3 | INCL | I | 1592 | |
| B4 | 52 | 00000000G | 8F | F3 | 000F5 | AOBLEQ | #LAST_DATA_LINE, YPOS, 13\$ | 1568 | |
| | 5A | 04 | A8 | 9E | 000FD | MOVAB | 4(R8), R10 | 1603 | |
| | | | 50 | D4 | 00101 | CLRL | R0 | | |
| | | | 6A | D5 | 00103 | TSTL | (R10) | | |
| | | | 04 | 12 | 00105 | BNEQ | 17\$ | | |
| | | | 50 | D6 | 00107 | INCL | R0 | | |
| | | | 0A | 11 | 00109 | BRB | 18\$ | | |
| | 03 | 00000000G | 00 | E9 | 0010B | BLBC | DISPLAYING, 18\$ | | |
| | | | 014D | 31 | 00112 | BRW | 32\$ | | |
| | 18 | | 50 | E9 | 00115 | BLBC | R0, 19\$ | 1608 | |
| 04 | AE | 00000000G | 8F | D0 | 00118 | MOVL | #FAOCTR_SIZE, FAOCSIZE | 1611 | |
| | | | 5A | DD | 00120 | PUSHL | R10 | 1612 | |
| | | 08 | AE | 9F | 00122 | PUSHAB | FAOCSIZE | | |
| 00000000G | 00 | | 02 | FB | 00125 | CALLS | #2, LIB\$GET_VM | | |
| | 01 | | 50 | E8 | 0012C | BLBS | STATUS, 19\$ | 1613 | |
| | | | | 04 | 0012F | RET | | | |
| | 56 | | 6A | D0 | 00130 | MOVL | (R10), POINTER | 1616 | |
| | | 42 | A8 | 95 | 00133 | TSTB | 66(R8) | 1618 | |
| | | | 0F | 13 | 00136 | BEQL | 20\$ | | |
| 03 | 44 | 50 | 00000000G | 00 | D0 | 00138 | MOVL | MRBPTR, R0 | |
| | | | 03 | E0 | 0013F | BBS | #3, 68(R0), 20\$ | | |
| | | | 0099 | 31 | 00144 | BRW | 28\$ | | |
| 09 | 4C | A8 | 03 | E1 | 00147 | BBC | #3, 76(R8), 21\$ | 1624 | |
| | | 50 | 00000000G | 8F | D0 | 0014C | MOVL | #WIDE_NAME_SIZE, COL_OFFSET | 1625 |
| | | | 07 | 11 | 00153 | BRB | 22\$ | | |
| | | 50 | 00000000G | 8F | D0 | 00155 | MOVL | #MAX_NAME_SIZE, COL_OFFSET | 1626 |
| | | 51 | 00000000G | 00 | 9A | 0015C | MOVZBL | NAME_COL, R1 | 1627 |
| 6E | | | 50 | C1 | 00163 | ADDL3 | COL_OFFSET, R1, XPOS | | |
| | | 41 | A8 | 94 | 00167 | CLRB | 65(R8) | 1628 | |
| | | 50 | 00000000G | 00 | D0 | 0016A | MOVL | MRBPTR, R0 | 1630 |
| 09 | 44 | A0 | 03 | E1 | 00171 | BBC | #3, 68(R0), 23\$ | | |
| | | 59 | 00000000G | 00 | 9E | 00176 | MOVAB | #SUMSTR, CUR_TABSTR | 1631 |
| | | | 14 | 11 | 0017D | BRB | 25\$ | | |
| | | 09 | 45 | A8 | E9 | 0017F | BLBC | 69(R8), 24\$ | 1632 |
| | | 59 | 00000000' | EF | 9E | 00183 | MOVAB | TABSTR_PC, CUR_TABSTR | 1633 |

| | | | | | | | | | |
|----|-----------|----|----|----|-------|--------|------------------------------|------|--|
| | | | 07 | 11 | 0018A | BRB | 25\$ | | |
| | | 59 | EF | 9E | 0018C | MOVAB | TABSTR, CUR TABSTR | 1634 | |
| 57 | 00000000G | 8F | 01 | C3 | 00193 | SUBL3 | #1, #FIRST_DATA_LINE, YPOS | 1636 | |
| | | | 39 | 11 | 0019B | BRB | 27\$ | | |
| | | 50 | A7 | 9E | 0019D | MOVAB | -1(R7), RO | 1639 | |
| 2D | 00000000' | EF | 50 | E1 | 001A1 | BBC | RO, SCR_DATA_LINE, 27\$ | | |
| | | 86 | 86 | 80 | 001A9 | MOVW | #22811, -(POINTER)+ | 1642 | |
| 86 | | | 57 | 81 | 001AE | ADDB3 | ROW OFFSET, YPOS, (POINTER)+ | 1643 | |
| | | 86 | 86 | 90 | 001B2 | MOVB | XPOS, (POINTER)+ | 1644 | |
| | | 50 | 69 | 9A | 001B5 | MOVZBL | (CUR TABSTR), RO | 1646 | |
| 66 | 01 | A9 | 50 | 28 | 001B8 | MOV3 | RO, T(CUR TABSTR), (POINTER) | | |
| | | 50 | 69 | 9A | 001BD | MOVZBL | (CUR TABSTR), RO | 1647 | |
| | | 56 | 50 | C0 | 001C0 | ADDL2 | RO, POINTER | | |
| | 00000000G | 8F | 57 | D1 | 001C3 | CMPL | YPOS, #FIRST_DATA_LINE | 1648 | |
| | | | 0A | 12 | 001CA | BNEQ | 27\$ | | |
| 50 | | 56 | 6A | C3 | 001CC | SUBL3 | (R10), POINTER, RO | 1649 | |
| 40 | A8 | 50 | A8 | 83 | 001D0 | SUBB3 | 65(R8), RO, 64(R8) | | |
| BF | | 57 | 8F | F3 | 001D6 | AOBLEQ | #LAST_DATA_LINE, YPOS, 26\$ | 1636 | |
| | | | 7E | 11 | 001DE | BRB | 31\$ | 1618 | |
| | | 57 | 1E | D0 | 001E0 | MOVL | #30, XPOSCOUNT | 1670 | |
| | | 58 | 27 | D0 | 001E3 | MOVL | #39, XPOSBAR | 1671 | |
| | | 86 | 8F | 80 | 001E6 | MOVW | #17947, (POINTER)+ | 1672 | |
| | 41 | A8 | 02 | 90 | 001EB | MOVB | #2, 65(R8) | 1674 | |
| 59 | 00000000G | 8F | 01 | C3 | 001EF | SUBL3 | #1, #FIRST_DATA_LINE, YPOS | 1676 | |
| | | | 58 | 11 | 001F7 | BRB | 30\$ | | |
| | | 50 | A9 | 9E | 001F9 | MOVAB | -1(R9), RO | 1679 | |
| 4C | 00000000' | EF | 50 | E1 | 001FD | BBC | RO, SCR_DATA_LINE, 30\$ | | |
| | | 86 | 86 | 80 | 00205 | MOVW | #22811, -(POINTER)+ | 1682 | |
| | | 86 | 59 | 90 | 0020A | MOVB | YPOS, (POINTER)+ | 1683 | |
| | | 86 | 57 | 90 | 0020D | MOVB | XPOSCOUNT, (POINTER)+ | 1684 | |
| 66 | 00000000' | EF | 09 | 28 | 00210 | MOV3 | #9, COUNTSTR, (POINTER) | 1686 | |
| | | 56 | 09 | C0 | 00218 | ADDL2 | #9, POINTER | 1687 | |
| | | 86 | 8F | 80 | 0021B | MOVW | #22811, (POINTER)+ | | |
| | | 86 | 59 | 90 | 00220 | MOVB | YPOS, (POINTER)+ | 1688 | |
| | | 86 | 5B | 90 | 00223 | MOVB | XPOSBAR, (POINTER)+ | 1689 | |
| 86 | 18 | 00 | EF | F0 | 00226 | INSV | REPTSTR, #0, #24, (POINTER)+ | 1691 | |
| | | 56 | 02 | C0 | 0022F | ADDL2 | #2, POINTER | 1692 | |
| | | 86 | 00 | 90 | 00232 | MOVB | BARCHAR, (POINTER)+ | | |
| | | 86 | 8F | 80 | 00239 | MOVW | #19227, (POINTER)+ | 1693 | |
| | 00000000G | 8F | 59 | D1 | 0023E | CMPL | YPOS, #FIRST_DATA_LINE | 1695 | |
| | | | 0A | 12 | 00245 | BNEQ | 30\$ | | |
| | | 50 | 6A | C3 | 00247 | SUBL3 | (R10), POINTER, RO | 1696 | |
| 40 | A8 | 50 | A8 | 83 | 0024B | SUBB3 | 65(R8), RO, 64(R8) | | |
| A0 | | 59 | 8F | F3 | 00251 | AOBLEQ | #LAST DATA LINE, YPOS, 29\$ | 1676 | |
| | | 86 | 8F | 80 | 00259 | MOVW | #18203, (POINTER)+ | 1701 | |
| 68 | | 56 | 6A | C3 | 0025E | SUBL3 | (R10), POINTER, (R8) | 1709 | |
| | | 50 | 00 | D0 | 00262 | MOVL | NORMAL, RO | 1711 | |
| | | | 04 | 00 | 00269 | RET | | 1712 | |

; Routine Size: 618 bytes, Routine Base: \$CODE\$ + 0000

```

585 1713 1 GLOBAL ROUTINE OUTPUT( STRING ) =
586 1714 2 BEGIN
587 1715 2
588 1716 2 ++
589 1717 2
590 1718 2 FUNCTIONAL DESCRIPTION:
591 1719 2
592 1720 2 Routine to output counted string with no carriage control.
593 1721 2
594 1722 2 INPUTS:
595 1723 2
596 1724 2 STRING - address of counted ascii string.
597 1725 2
598 1726 2 OUTPUTS:
599 1727 2
600 1728 2 none
601 1729 2 ++
602 1730 2
603 1731 2 PUT_TO_SCREEN (.(.STRING)<0,8>, .STRING+1)
604 1732 1 END;

```

```

7E      04      AC      0000 00000
          7E      04      01  C1 00002
          00000000G 00      BC  9A 00007
                                02  FB 0000B
                                04 00012

```

```

.ENTRY OUTPUT, Save nothing
ADDL3 #1, STRING, -(SP)
MOVZBL @STRING, -(SP)
CALLS #2, PUT_TO_SCREEN
RET

```

```

: 1713
: 1731
:
: 1732

```

; Routine Size: 19 bytes, Routine Base: \$CODE\$ + 026A

```

605 1733 1
606 1734 1
607 1735 1 ROUTINE POSITION( YPOS , XPOS ) =
608 1736 2 BEGIN
609 1737 2
610 1738 2 ++
611 1739 2
612 1740 2 FUNCTIONAL DESCRIPTION:
613 1741 2
614 1742 2 Routine to call SCRPKG to position screen for characters.
615 1743 2
616 1744 2 INPUTS:
617 1745 2
618 1746 2 YPOS - y position ( row number , one origin)
619 1747 2 XPOS - x position ( column number , one origin)
620 1748 2
621 1749 2 OUTPUTS:
622 1750 2
623 1751 2 none
624 1752 2 --
625 1753 2
626 1754 2 SCR$SET_CURSOR (.YPOS, .XPOS) ! set cursor to the requested position
627 1755 1 END;

```


| | | | | | | | |
|-----------|----|----|----|----------------------|-------|----------------------|--------|
| | | | | 0000 00000 POSITION: | | | |
| | | 7E | 04 | AC 7D 00002 | .WORD | Save nothing | : 1735 |
| 00000000G | 00 | | | 02 FB 00006 | MOVQ | YPOS, -(SP) | : 1754 |
| | | | | 04 0000D | CALLS | #2, \$CR\$SET_CURSOR | : 1755 |
| | | | | | RET | | |

; Routine Size: 14 bytes, Routine Base: \$CODE\$ + 027D

| | | | |
|-------|------|---|--------|
| : 628 | 1756 | 1 | |
| : 629 | 1757 | 1 | |
| : 630 | 1758 | 1 | END |
| : 631 | 1759 | 0 | ELUDOM |

!End of module

| PSECT SUMMARY | | | | | | | | | |
|---------------|-------|-------|-------|------------|-------|-------|-----|-----|--------------------|
| Name | Bytes | | | Attributes | | | | | |
| \$OWNS | 147 | NOVEC | WRT | RD | NOEXE | NOSHR | LCL | REL | CON,NOPIC,ALIGN(2) |
| \$PLITS | 436 | NOVEC | NOWRT | RD | NOEXE | NOSHR | LCL | REL | CON,NOPIC,ALIGN(2) |
| \$CODE\$ | 651 | NOVEC | NOWRT | RD | EXE | NOSHR | LCL | REL | CON,NOPIC,ALIGN(2) |
| ABS | 0 | NOVEC | NOWRT | NORD | NOEXE | NOSHR | LCL | ABS | CON,NOPIC,ALIGN(0) |

| Library Statistics | | | | | | |
|---------------------------------|-------|----------------|---------|--------------|-----------------|--|
| File | Total | Symbols Loaded | Percent | Pages Mapped | Processing Time | |
| _\$255\$DUA28:[SYSLIB]LIB.L32;1 | 18619 | 5 | 0 | 1000 | 00:01.9 | |

COMMAND QUALIFIERS

BLISS/CHECK=(FIELD,INITIAL,OPTIMIZE)/LIS=LIS\$:TEMPLATE/OBJ=OBJ\$:TEMPLATE MSRC\$:TEMPLATE/UPDATE=(ENH\$:TEMPLATE)

Size: 651 code + 583 data bytes

Run Time: 00:33.4

Elapsed Time: 01:07.0

Lines/CPU Min: 3157

TEMPLATE
V04-000

⁶
16-Sep-1984 02:18:37

VAX-11 Bliss-32 V4.0-742

Page 23

: Lexemes/CPU-Min: 40116
: Memory Used: 351 pages
: Compilation Complete

_S

Sy

GE
GE

GE
GE
GE
GE
GL
GL
HE

HO
HO

IN
IN
IN
IO

IO
IO
IO

IO
IO
IO
IO
IO
IO
IO
IO
JR
JR
JR
JR
KE
KE
LA
LA

LA
LA
LA
LC
LC
LE
LI
LI
LI
LI

0243 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

